UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/667,756	09/22/2003	Udayan Rajendra Kanade	COT-004	1373

7590 07/22/2008

William L, Botjer
PO Box 478
Center Moriches, NY 11934

EXAMINER	
KENDALL, CHUCK O	

ART UNIT	PAPER NUMBER
2192	

MAIL DATE	DELIVERY MODE
07/22/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/667,756	KANADE, UDAYAN RAJENDRA	
	Examiner	Art Unit	
	CHUCK O. KENDALL	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *22 September 2003*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-27* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-27* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *22 September 2003* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

Detailed Action

1. This is in response to Application filed 09/22/03.

2. Claims 1 – 27 have been examined.

Claim Rejections - 35 USC § 101

3. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

4. Claims 1 - 2 and 17 – 27 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter. Claims appears to draw limitations merely software per se limitations. There doesn't appear to be any interrelated associated hardware which would enable the claimed functionality to be achieved.

Claim Rejections - 35 USC § 103

5. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

6. Claims 1 – 27 are rejected under 35 U.S.C. 103(a) as being unpatentable over Doing et al. US 6175916 in view of Ginsberg US 6175916.

Regarding claims 1, 3, 17,19, 24, 26 and 27 a floating thread structure for programming threads within a multithreaded application program, the floating thread structure minimizing thread switching overheads and memory usage during execution of the multithreaded application program, the floating thread structure comprising :

a. a main level function, the main level function being the basic execution level function for the thread, the main level function making subsequent calls to all other functions (Doing, 4:60 – 5:20).

Doing doesn't expressly disclose wherein the sub-functions following a function calling convention, the sub-functions comprising, non-preemptive functions, the non-preemptive functions being normal functions that do not stall the thread at any stage of execution, the non-preemptive functions being called from the main level function or any sub-function called by the main level function, the non-preemptive functions being capable of making calls to other non-preemptive functions only, preemptive functions, the preemptive functions being functions that may cause a thread to temporarily stall further execution, the preemptive functions being called from the main level function only, the preemptive functions being capable of making calls to other non-preemptive functions only and other standard program constructs relevant to the thread. However, Ginsberg discloses in an analogous art and similar configuration discloses, in 7:1 – 10, that,

"... The call handler has a non-preemptible portion, referred to as a memory fault handler, that takes
care of critical functions such as changing virtual memory mappings. It also has a preemptible portion,
referred to as a call processing function, that performs less critical tasks as described below…"

Therefore it would have been obvious to one of ordinary skill in the art at the time the invention was made to combine Doing and Ginsberg because, it would enable the system to free up memory resources as suggested by Ginsberg in 7:1 – 10.

Regarding claim 4, the method as recited in claim 3 wherein the preemptive function is written in a manner that the function restarts from its beginning if it preempts the thread (Doing, 13:60 – 65, see restart).

Regarding claim 5, the method as recited in claim 4 wherein writing the preemptive function further comprises:

a. ascertaining the state of the thread, the thread state being ascertained by checking the value of the condition code for the thread, the condition code being a special field associated with the thread, the value stored in the condition code field being indicative of the thread state (Doing, 13:60 – 67),

b. performing application specific logic in accordance with the preemptive function(Doing, 13:60 – 67);

c. ascertaining whether the function needs to preempt the thread, the ascertaining being based upon the condition code and other application specific logic (Doing, 13:60 – 67); and

d. preempting the thread in case the function needs to preempt the thread, the preemption being done after setting the condition code to an appropriate value and performing other application specific logic (Doing, 13:60 – 67),

Regarding claim 8, the method as recited in claim 7 wherein compiling the threads in accordance with the function calling format comprises:

a. compiling a main level function, the main level function being the basic execution level function of the thread, the main level function making all subsequent calls to other functions (Doing, 18:50 – 65);

b. compiling non-preemptive functions, non-preemptive functions being normal functions that do not stall the thread at any stage of execution, the non-preemptive functions being called from the main level function or any sub-function called by the main level function, the non-preemptive functions being capable of making calls to other non-preemptive functions only (18:50 – 65);

c. compiling preemptive functions, the preemptive functions being those functions that may cause a thread to temporarily stall and yield control to the operating system, the preemptive functions being called from the main level function only, the preemptive functions being capable of making calls to other non-preemptive functions (18:50 – 65); and

d. compiling other program constructs (18:50 – 65).

Regarding claim 9, the method as recited in claim 8 wherein compiling the main level function comprises:

a. providing instructions for calling a preemptive function comprising:

i. providing instructions for storing and updating operations in response to the preemptive function call, the operations being performed so that the necessary context information pertaining to the thread is not lost if the thread is preempted during the preemptive function call (Ginsberg, 7:1 – 10);

ii. providing instructions for enabling the preemptive function to distinguish between the first and subsequent calls to it, the subsequent calls being made in case the preemptive function preempts the thread(Ginsberg, 7:3 – 20, see call handler);

iii. providing instructions for facilitating switching back of the thread, the switching back being necessary in case the preemptive function call preempts the thread, the facilitating being done in such a manner that the thread is restarted from the beginning of the preemptive function during a subsequent call (Doing, 5:35 – 45, see switch logic and 13:60 – 65, see restart); and

iv. providing instructions for calling the preemptive function with appropriate parameters; and

b. providing instructions for other program constructs and non-preemptive function calls(Ginsberg, 7:1 – 10).

Regarding claim 10, the method as recited in claim 9 wherein providing instructions for storing and updating operations in response to a preemptive function call comprises:

a. providing instructions for storing all live local and temporary variables on local memory, the live variables being those variables that have been defined and assigned a value prior to the preemptive function call, the values so defined being used after the preemptive function call (Doing, 11:30 – 45,see livelock issues and incorporation by reference); and

b. providing instructions for loading the values assigned to the live variables from the local memory when they are required for subsequent execution of the main level function (Doing, 11:30 – 45,see livelock issues).

Regarding claim 11, the method as recited in claim 9 where providing instructions for enabling the preemptive function to distinguish between its first and subsequent calls comprises providing instructions for initializing the thread's condition code, the condition code being a special field associated with the thread used for ascertaining the thread state, the initial value assigned to the field being indicative of the fact that the thread has not been preempted as yet due to the ensuing instance of the preemptive function (Ginsberg, 7:1 – 10).

Regarding claim 12, Ginsberg further discloses the method as recited in claim 9 wherein providing instructions for facilitating the switching back of the thread comprises:

a. providing instructions for setting called function pointer of the thread to point to the preemptive function being called, the called function pointer being a special pointer associated with the thread, the pointer being subsequently used to point at the function that caused the thread to preempt (Ginsberg,6:45 – 55);

b. providing instructions for saving parameters related to the function call in memory (6:45 – 55); and

c. providing instructions for setting up an executing stack for the preemptive function, the execution stack being needed for maintaining the activation records related to the preemptive function, the setting up of the stack being done in a manner such that an equivalent stack can be set up during switching back of the thread (6:45 – 55).

Regarding claim 13, the method as recited in claim 8 wherein compiling the preemptive function call comprises providing instructions for using the same stack as used by the main level function, the stack being overwritten during the preemptive function call, overwriting of the stack having been facilitated by storing all pertinent information residing on the stack during compilation of main level function (9:35 – 50).

Regarding claim 14, the method as recited in claim 7 wherein executing the compiled thread comprises:

a. setting up the thread for running (Doing, FIGURE 6 and all associated text);

b. running the thread by executing compiled code (Doing, FIGURE 6 and all associated text and 18:50 – 65 for compiling);

c. preempting the thread and switching to other threads when the thread requests preemption, the preemption being done without saving any context information, the context information being pertinent information associated with the thread (5:35 – 45, see thread switch logic); and

d. restarting the preempted thread when the thread is scheduled back for running, such restarting being done with a minimum amount of loading of context information (Doing, 13:60 – 65, see restart).

Regarding claims 15 and 23, the method as recited in claim 14 wherein the step of setting up the thread comprises:

a. setting up the execution stack for the activation records of the main level function and other functions within the thread, the thread stack capable of being overwritten during the execution of the thread (Ginsberg, 4:50 – 55); and

b. calling the main level function of the thread(Ginsberg, 4:50 – 55 and 7:1 – 10).

Regarding claim 16, the method as recited in claim 14 wherein the step of restarting the thread comprises:

a. setting up the execution stack for execution of the preemptive function that caused the thread to preempt, the setting up being done in a manner such that the subsequent execution of the thread may continue on the same stack, the setting up being done to make minimal use of memory(7:1 – 10).; and

b. calling the preemptive function that caused the thread to preempt, the call being made from the beginning of the function, the calling being facilitated by the main level function having saved all information pertinent to the restarting of the preemptive function (7:1 – 10).

Regarding claim 18, the system as recited in claim 17 wherein the compiler service compiling the application program comprises:

a. a compiler compiling normal threads in accordance with the standard methodology for compiling threads(Doing, 18:50 – 65); and

b. a floating thread compiler compiling floating threads in a manner such that no reference information needs to be stored on the processor when the floating thread is swapped out of the processor and subsequently swapped back into the processor(Doing, 18:50 – 65).

Regarding claim 20, the system as recited in claim 17 wherein the memory allocated to normal and floating threads comprises:

a. a plurality of normal thread stacks, each of the normal thread stacks being associated with a normal thread, the stack being stored with context information pertaining to the thread, the context information being the entire information set relevant to the thread (Doing, FIGURE 4b, and all associated text);

b. a plurality of floating thread stores, each of the thread stores being associated with a floating thread, the thread stores being stored with minimal context information that needs to be kept persistent across a floating thread switch (FIGURE 6, and all associated text); and

c. a floating thread stack associated with the processor, the floating thread stack being used by the floating thread being processed by the processor (FIGURE 6, and all associated text).

Regarding claim 21, system as recited in claim 17 wherein the multithreaded processing system works in a multiprocessor environment, each of the multiple processors having a separate floating thread stack associated with it (Doing, 5:35 – 45).

Regarding claim 22, the system as recited in claim 17 wherein the operating system managing the scheduling of threads comprises:

a. a scheduler ready queue holding threads that are ready for execution (Ginsberg 2:13 – 30);

b. a normal thread preemption module providing preemptive services to normal threads, the preemptive services being specific activities that might preempt a thread (Ginsberg, 7:1 – 10) and

c. a floating thread preemption module providing preemptive services to floating threads (Ginsberg, 7:1 – 10).

Regarding claim 25, the floating thread compiler as recited in claim 24 further comprising:

a. means for identifying the main level, preemptive and non-preemptive function blocks within the floating thread (Doing, 4:60 – 5:20); and

b. means for activating the appropriate function compiler for compiling each of the functions(18:50 – 65).

Claim Rejections - 35 USC § 102

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

6. Claim 7 is rejected under 35 U.S.C. 102(b) as being anticipated by Doing et al. US 6175916.

Regarding claim 7, Doing anticipates a method for minimizing thread switching overheads and number of execution stacks used by threads in multithreaded processing, the method comprising:

a. compiling the threads in accordance with a function calling format, the function calling format ensuring that the thread need not save large context information during thread switching (18:40 – 50); and

b. executing the compiled application code (18:40 – 50).

Allowable Subject Matter

7. Claims 2 and 6 are objected to as being dependent upon a rejected base claim, but would be allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims.

The following is not expressly taught in anyone prior art or combination of prior art:

"… wherein each function definition is preceded by a keyword declaration, the keyword declaration distinguishing amongst main level function, preemptive functions and non-preemptive functions. However, Wolf in an analogous art and similar configuration discloses..."

Correspondence information

8. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Chuck Kendall whose telephone number is 571-272-3698. The examiner can normally be reached between Monday and Thursday, at 11:00 am - 4:300pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is **571-273-8300**.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Chuck O Kendall/

Primary Examiner, Art Unit 2192